PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In Re Application of:	)	
	)	
Ahluwalia, et al.	)	Confirmation No. 1055
	)	
Serial No.: 10/790,509	)	Examiner: Li, Zhuo H.
	)	Group Art Unit: 2185
Filed: March 1, 2004	)	
	)	
For: **Memory Management**	)	HP Docket No.: 20035654-1
	)	TKHR Docket No.: 050849-1450
	)	

APPEAL BRIEF UNDER 37 C.F.R. § 41.37

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed herewith, responding to the final Office Action mailed April 29, 2009.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest of the instant application is Hewlett-Packard Development Company, a Texas Limited Liability Partnership having its principal place of business in Houston, Texas.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF THE CLAIMS

Claims 1-23 are pending in this application. Claims 1-23 stand rejected by the final Office Action, and are the subject of this appeal.

IV. STATUS OF AMENDMENTS

There have been no claim amendments made after the final Office Action, and all amendments made before the final Office Action have been entered. The claim listing in section VIII (CLAIMS – APPENDIX) represents the present state of the claims.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Embodiments of the claimed subject matter are summarized below with reference numbers and references to the written description ("specification") and drawings. The subject matter described below appears in the original disclosure at least where indicated, and may further appear in other places within the original disclosure.

Embodiments according to independent claim 1 involve a computing device, comprising: a processor (p. 5 line 22 to p. 6 line 5; 107 in FIG. 1); a memory coupled to the processor (p. 5 lines 22-29; p. 6 lines 8-15; 130 in FIG. 1); and program instructions provided to the memory and executable by the processor (p. 16 lines 18-20) to: track a virtual address space for a process associated with a device connected to the computing device (p. 9 lines 12-14; p. 12 line 29 to p. 13 line 16; p. 13 lines 28-31; p. 14 liens 11-15; p. 16 lines 1-6; 510 in FIG. 5); release a physical address space associated with the virtual address space when the device has a

connection removed from the computing device (p. 4 lines 27-31; p. 5 lines 3-7; p. 16 lines 7-9 and lines 27-30; 520 in FIG. 5); provide an indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid for use by the process (p. 4 line 27 to p. 5 line 2; p. 16 lines 2732; p. 17 lines 9-14; p. 18 lines 13-17; p. 19 lines 6-10 and 21-33; p. 20 lines 19-26; 530 in FIG. 5); wherein the indication is triggered by detection that the physical address space that was being used by processes associated with the device has been released (p. 20 line 21 to p. 21 line 2); and wherein the indication occurs responsive to the physical address space being released and before release of the virtual address space by the process (p. 20 line 30 to p. 21 line 2; p. 23 lines 16-20).

Embodiments according to independent claim 8 involve a computing device, comprising: a processor (p. 5 line 22 to p. 6 line 5; 107 in FIG. 1); a random access memory coupled to the processor (p. 5 lines 22-29; p. 6 lines 8-15; 130 in FIG. 1); and program instructions provided to the memory and executable by the processor (p. 16 lines 18-20), the program instructions are part of a memory management system (p. 6 lines 8-20; 135 in FIG. 1) to: dereference a virtual address space for a process associated with a removable memory mappable device connected to the computing device (p. 11 lines 25-28; p. 16 lines 22-27; p. 18 lines 1-6; 410 in FIG. 4); release a physical address space associated with the virtual address space when the device associated with the process is logically disconnected (p. 4 lines 27-31; p. 5 lines 3-7; p. 16 lines 7-9 and lines 27-30; 520 in FIG. 5); and register by providing an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process (p. 4 line 27 to p. 5 line 2; p. 16 lines 27-32; p. 17 lines 9-14; p. 18 lines 13-17; p. 19 lines 6-10 and 21-33; p. 20 lines 17-26; 530 in FIG. 5); wherein to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released (p. 9 lines 12-14; p. 20 line 21 to p. 21 line 2); and wherein to register

occurs as the physical address space is released and before release of the virtual address space by the process (p. 20 line 30 to p. 21 line 2; p. 23 lines 16-20).

Embodiments according to independent claim 13 involve a computing device, comprising: a processor (p. 5 line 22 to p. 6 line 5; 107 in FIG. 1); a memory coupled to the processor (p. 5 lines 22-29; p. 6 lines 8-15; 130 in FIG. 1), the memory including program instructions for maintaining a virtual memory data structure (250 in FIG. 2) specific to a process (p. 17 lines 9-14) as part of a memory management system (p. 6 lines 8-20; 135 in FIG. 1); and means for unmapping a virtual address space for the process that is triggered as a physical address space used by the process is being released, in a manner which does not violate semantics for an operating system of the computing device, when a removable memory mappable device associated with the process is logically disconnected (p. 4 lines 24-31; p. 5 lines 10-15; p. 17 lines 7-19; p. 18 lines 16-23).

Embodiments according to independent claim 19 involve a method for memory management on a computing device, comprising: dereferencing a memory address for a process associated with a removable memory mappable device (p. 11 lines 25-28; p. 16 lines 22-27; p. 18 lines 1-6; 410 in FIG. 4); mapping a representation of an object associated with the process in a virtual memory data structure associated with the process (p. 17 lines 4-8; p. 18 lines 6-10; p. 420 in FIG. 4); removing the object from physical memory when the device is logically disconnected from the computing device (p. 17 lines 4-8; p. 18 lines 10-12; 430 in FIG. 4); and providing an indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process as triggered by detection of a physical address space used by the process being released and when the object is removed from physical memory (p. 4 line 27 to p. 5 line 2; p. 13 lines 13-15; p. 16 lines 2732; p. 17 lines 9-14; p. 18 lines 13-17; p. 19 lines 6-10 and 21-33; p. 20 lines 19-26; 440 in FIG. 4; 530 in

FIG. 5)), without removing the representation of the object from the virtual memory data structure for the process (p. 18 lines 16-20; 440 in FIG. 4).

Embodiments according to independent claim 22 involve a method for memory management, comprising: tracking a virtual address space for a process associated with a removable memory mappable device connected to a computing device (p. 9 lines 12-14; p. 12 line 29 to p. 13 line 16; p. 13 lines 28-31; p. 14 liens 11-15; p. 16 lines 1-6; 510 in FIG. 5); releasing a physical address space when the device has a logical connection removed from the computing device (p. 4 lines 27-31; p. 5 lines 3-7; p. 16 lines 7-9 and lines 27-30; 520 in FIG. 5); and at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is not available to the process (p. 4 line 27 to p. 5 line 2; p. 9 lines 12-14; p. 16 lines 27-32; p. 17 lines 9-14; p. 18 lines 13-17; p. 19 lines 6-10 and 21-33; p. 20 lines 17-26; p. 20 line 21 to p. 21 line 2; p. 20 line 30 to p. 21 line 2; p. 23 lines 16-20; 530 in FIG. 5) in a manner which does not violate semantics of an operating system (p. 4 lines 24-31; p. 5 lines 10-15; p. 17 lines 7-19; p. 18 lines 16-23).

Embodiments according to independent claim 23 involve a computer readable storage medium having computer readable instructions stored thereon for execution by a device (p. 7 lines 8-17) to perform a method, comprising: dereferencing a virtual address space for a process associated with a removable memory mappable device as part of a memory management system on a computing device; releasing a physical address space when the device is logically disconnected from the computing device (p. 4 lines 27-31; p. 5 lines 3-7; p. 16 lines 7-9 and lines 27-30; 520 in FIG. 5); and at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process (p. 4 line 27 to p. 5 line 2; p. 9 lines 12-14; p. 16 lines 27-32;

p. 17 lines 9-14; p. 18 lines 13-17; p. 19 lines 6-10 and 21-33; p. 20 lines 17-26; p. 20 line 21 to p. 21 line 2; p. 20 line 30 to p. 21 line 2; p. 23 lines 16-20; 530 in FIG. 5) in a manner which does not violate semantics for an operating system the computing device (p. 4 lines 24-31; p. 5 lines 10-15; p. 17 lines 7-19; p. 18 lines 16-23).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are to be reviewed on appeal.

A. Claim 23 stands rejected under 35 U.S.C. § 112, ¶1, as failing to comply with the written description requirement.

B. Claims 1-23 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over *Armilli* (U.S. 6,907,494) in view of *Browning et al.* (U.S. 6,918,023).

VII. ARGUMENTS

A. Rejection of Claim 23 under 35 U.S.C. § 112, ¶1 (written description)

1. The *prima facie* case is insufficient

The Office Action alleges that:

> "[A] computer readable storage medium" is not properly defined in the specification. It appears that the specification merely define a computer readable medium may include any medium that can store or transfer information (see page 7 line 8-21). The original specification fails to clearly define or describe what is "a computer readable storage medium" in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.
> (Office Action, p. 2.)

Appellant agrees with the premise of the argument – that the instant specification defines a computer readable medium as including any medium that can store or transfer information – but submits that the conclusion drawn by the Office Action is not supported by this premise.

Specifically, the Examiner's statement implies that the specification does not clearly define ***computer readable storage medium*** simply because it contains a broad definition of ***computer readable medium***. The Examiner appears to ignore the adjective "storage", which further defines, and narrows from, the definition of "medium". Thus, the fact that "computer

readable medium" is defined broadly does not lead to the conclusion that "computer readable storage medium" is impermissibly broad so as to show that Appellant did not possess the claimed invention. Thus, Appellant submits that the Examiner's *prima facie* case for a written description rejection is deficient, and the rejection should be overturned.

2. The specification does reasonably convey possession of a "computer readable storage medium"

Appellant now turns to the actual issue of whether the specification reasonably conveys to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed "computer readable storage medium". The Office Action (p. 2) objects that the "original specification fails to *clearly define* or describe what is 'a computer readable storage medium' ". (Emphasis added). Appellant submits that no description is necessary because the term is clearly understood, as a matter of plain English, to be a particular type of computer readable medium, one that stores computer readable instructions. In fact, the specification specifically supports this understanding: "A computer readable medium may include any medium that can store or transfer information." (para. 0024.)

However, the specification does give additional detail, by giving the following examples: "electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable ROM (EROM), a floppy diskette, a compact disk CD-ROM, an optical disk, a hard disk..." Appellant submits that a person of ordinary skill in the art would understand each of these to be a storage medium. No further description of a computer readable storage medium is necessary because such media are well known to a person of ordinary skill in the art. As noted in MPEP 2163, what is conventional or well known to one of ordinary skill in the art need not be disclosed in detail. See *Hybritech Inc. v. Monoclonal Antibodies, Inc.*, 802 F.2d at 1384, 231 USPQ at 94. See also *Capon v. Eshhar*, 418 F.3d 1349, 1357, 76 USPQ2d 1078, 1085 (Fed. Cir. 2005)("The 'written description' requirement must be applied in the context of the particular invention and the state of the knowledge.. As each field evolves, the balance also evolves between what is known and

what is added by each inventive contribution."). If a skilled artisan would have understood the inventor to be in possession of the claimed invention at the time of filing, even if every nuance of the claims is not explicitly described in the specification, then the adequate description requirement is met. *See, e.g.*, *Vas-Cath*, 935 F.2d at 1563, 19 USPQ2d at 1116; *Martin v. Johnson*, 454 F.2d 746, 751, 172 USPQ 391, 395 (CCPA 1972) (stating "the description need not be in *ipsis verbis* [i.e., "in the same words"] to be sufficient").

3. The Examiner unfairly replaced a § 101 rejection with a § 112, ¶1 (written description) rejection

Appellant notes that the term at issue in this written description rejection ("storage") was added to the claim in response to a suggestion by the Examiner as to how Appellant could overcome a rejection under 35 U.S.C. § 101. (See Office Action mailed November 24, 2008, pp. 1-2.) In this § 101 rejection, the Examiner alleged that without the "storage" qualifier, the medium was broad enough to encompass intangible embodiments. Appellant amended as indicated, in reliance on the Examiner's suggestion, only to find that the Examiner then replaced the § 101 rejection with the current § 112, ¶1 (written description) rejection.

4. Conclusion

For at least these reasons, the rejection under § 112, ¶1 written description is clearly misplaced and improper, and should be overturned.

B. Rejection of Claims 1-23 under 35 U.S.C. § 103(a): *Armilli* and *Browning et al.*

Appellant submits that a *prima facie* case of obviousness for claims 1-23 has not been established using the references of record, for at least the following reasons. Therefore, Appellant requests that the rejection be overturned.

The U.S. Patent and Trademark Office bears the burden under 35 U.S.C. §103 to establish obviousness. *In re Fine*, 837 F.2d 1071, 1074, 5 U.S.P.Q. 2d 1596, 1598 (Fed. Cir. 1988). A proper rejection of a claim under 35 U.S.C. §103 as being obvious based upon a combination of references requires that the cited combination of references must disclose, teach, or suggest

(either implicitly or explicitly) all elements/features/steps of the claim at issue. *See, e.g., In re Dow Chemical*, 5 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1988); *In re Keller*, 208 U.S.P.Q.2d 871, 881 (C.C.P.A. 1981). Furthermore, even if the combination discloses all the elements, "rejections on obviousness cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418, 82 USPQ2d 1385, 1396 (quoting *In re Kahn*, 441 F.3d 977, 988, 78 USPQ2d 1329, 1336 (Fed. Cir. 2006)) (2007).

1. Independent Claim 1

a. The proposed combination does not teach "provide an indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid for use by the process"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations. associated with the process

(1) RPN list processing in *Browning et al.* does not correspond to "indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid"

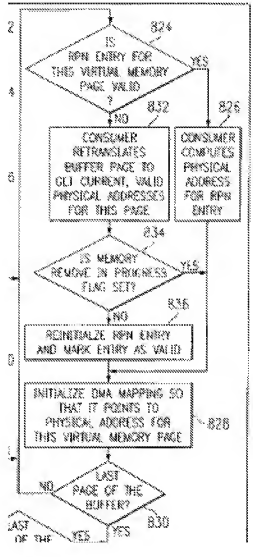
The Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches…the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., ***scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed***), and (col. 8 lines 53-58, i.e., ***receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed***).
> (Office Action, p. 4, emphasis added.)

Appellant disagrees with this allegation.

Appellant notes that the relied-upon portion of FIG. 8 (reproduced below) uses the term "valid" (steps 824, 836, and 914) when referring to an RPN data structure. Appellant also notes that the claim refers to a "virtual memory data structure" and includes the functional language "that the virtual address space, previously available to the process, is no longer valid". Thus, the Office Action appears to rely on the RPN for teaching the claimed "virtual memory data structure".

FIG. 8



Browning et al. does disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN. (See FIG. 3A.) Appellant therefore assumes (for the sake of argument) that the RPN is a "virtual memory data structure". Even so, *Browning et al.* does not teach or suggest that invalidating an RPN entry provides an "indication…that the virtual address space, previously available to the process, is no longer valid" as recited in claim 1.

Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page to get current, valid physical addresses.)

In addition, *Browning et al.* does not teach or suggest that the RPN is a data structure "associated with the process" as recited in claim 1. In fact, *Browning et al.* appears to suggest that the RPN is not associated with a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the ***lists are not owned by any particular entity***. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13, emphasis added.)

(2) In-progress flag in *Browning et al.* does not correspond to "indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid"

As discussed above, the Office Action specifically asserts that invalidating the RPN in *Browning et al.* corresponds to the claimed indication. However, in the Response to Arguments section the Office Action changes position and appears to rely on clearing the memory-move-in-progress flag in *Browning et al.* for teaching the "associated with the process" portion of the indication:

> In respond to Appellant's argument that the proposed combination does not teach "provide an indication in a virtual memory data structure associated with the process", Browning clearly teaches providing a flag to indicate whether virtual memory is removed in progress (step 834, figure 8 and col. 8 lines 32-44) and reinitializing RPN entry and mark entry as valid when memory remove in progress flag is not set ***such that the memory remove in progress flag provides an indication in a virtual memory data structure associated with the process***.
> (Office Action, p. 12, emphasis added.)

Appellant disagrees with this allegation. First, *Browning et al.* does not teach or suggest that the memory-remove-in-progress flag is part of a virtual memory data structure. Second, *Browning et al.* does not teach or suggest that the flag has anything to do with a particular process as required by claim 1.

(3) *Browning et al.* as a whole does not teach "indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid"

As noted above, the Office Action appears to take the position that updating one data structure (the memory-remove-in-progress flag) teaches one isolated portion ("indication in a virtual memory data structure associated with the process") of the claimed indication, and that updating a different data structure (the RPN entry) teaches the remaining part ("indication…that the virtual address space, previously available to the process, is no longer valid"). Yet the Examiner has provided no rationale for why these two separate data structures would be combined in the manner described in claim 1. The Examiner has failed to consider the claims as a whole. Appellant submits that it is not enough that all the individual elements of a claim are disclosed; those elements must also be arranged as in the claim.

(4) The combination as a whole does not teach "indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid"

Appellant has explained, in sections VII.B.1.a(1)-VII.B.1.a(3), why *Browning et al.* does not teach this feature. The Office Action acknowledges (pp. 3-4) that *Armilli* does not teach this feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

b. The proposed combination does not teach "the indication is triggered by detection that the physical address space that was being used by processes associated with the device has been released"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations.

(1) Checking the memory-move-in-progress flag in *Browning et al.* does not correspond to "the indication is triggered by detection that the physical address space that was being used by processes associated with the device has been released"

The Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), ***when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering the indication is triggered by detection that the physical address space that was being used by process associated with the device is being released,*** and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed), and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, p. 4, emphasis added.)

Appellant respectfully disagrees with this allegation.

The Office Action specifically refers to steps 824 and 834 of FIG. 8 for this teaching. With regard to step 824, Appellant respectfully submits that checking the validity of an RPN entry is not the same as detecting the release of physical addresses. Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid. (See FIG. 8, blocks 824 and 832.) Even assuming that checking validity of an RPN is the same as detecting release of physical addressees, the relied-upon portion of *Browning et al.* teaches that the action taken on detection is to update the virtual-to-physical mapping, rather than to trigger an indication as recited in claim 1.

With regard to step 834, Appellant respectfully submits that checking the memory-remove-in-progress flag set flag in step 834 is not the same as detecting that the physical address space has been released. Instead, this flag merely indicates that process 900 (FIG. 9) is executing. Furthermore, claim 1 requires more than such a detection, it requires a specific action. The action that occurs in *Browning et al.* if the memory-remove-in-progress condition is detected is initialization of DMA mapping, not triggering an indication of any kind, much less the specific indication recited in claim 1, namely an "indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid".

(2) In-progress flag in *Browning et al.* does not correspond to "the indication is triggered by detection that the physical address space that was being used by processes associated with the device has been released"

As discussed, above the Office Action specifically asserts that this feature corresponds to checking a particular condition (validity of RPN or the value of memory-move-in-progress flag) in *Browning et al.* However, in the Response to Arguments section the Office Action changes position and appears to rely instead on clearing the memory-move-in-progress flag:

> In respond to Appellant's argument that the proposed combination does not teach "wherein registering the indication is triggered by detection the physical address space…has been released", ***Browning teaches to clear memory remove in progress flag when it is acknowledge that real pages that are within range of memory has been removed or release*** (figure 9 and col. 8 line 65 through col. 9 line 10). Thus, Browning does teach the claimed limitations as recited in claim 1.
> (Office Action, p. 13, emphasis added.)

Appellant disagrees with this allegation.

First, Appellant submits that the "memory remove operation" referred to in FIG. 9 is not the same as the release of physical address space as described in claim 1. Instead, the "remove" is actually a relocate from one physical address to another. This is made clear in another portion of *Browning et al.*:

> In this manner, many different copies of a translation of a virtual address to physical address may exist. *Some of these pretranslations may be invalidated,*

> *such as when the physical address changes because a real page is* ***migrated*** *to a new real page.* When this occurs, the pretranslations to the original real page are invalid. The present invention provides a method, system, and product for locating particular pretranslations, invalidating them, synchronizing the invalidation process with the memory remove process, and then repopulating these lists with the current, valid pretranslation.
> (Col. 3, lines 17-25.)
>
> In order to *synchronize invalidation of pretranslations stored in these lists with a* ***memory remove*** *operation*, a user of a pretranslation list first disables the user's processor's ability to respond to interprocessor interrupts. The user then accesses the list. Once the user has finished accessing the list, the user then re-enables the ability of its processor to respond to interprocessor interrupts. Thus, while the user is accessing a pretranslation list, the user's processor will not respond to interprocessor interrupts.
> (Col. 3, lines 35-45.)

The synchronization using interprocessor interrupts described in this passage is the same synchronization described in connection with FIG. 9, leading to the conclusion that the migrate-real-page operation in the above passage is the same as the memory-remove operation of FIG. 9. Thus, the relied-upon passage (FIG. 9 and corresponding text) does not teach "physical address space…has been released", but rather that it has been moved.

Second, even assuming that the relied-upon passage teaches removing physical address space, it does not teach that this physical address space "was being used by processes associated with the device" as required by claim 1. FIG. 9 of *Browning et al.* does not describe processes or a particular device in the context of moving physical pages.

Third, even assuming that detecting the kernel has completed migration of real pages teaches detecting that physical address space has been released, it does not teach that such detection triggers an indication of any kind. According to FIG. 8, the action that occurs when the memory-remove-in-progress flag is clear (step 834) is marking the RPN entry as valid and reinitializing the entry (step 836). Appellant submits that changing values in an RPN entry is not the same as triggering an indication of kind, much less the specific indication recited in claim 1, namely an "indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid".

(3) The combination as a whole does not teach "an indication in a virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for use by the process"

Appellant has explained, in sections VII.B.1.b(1) and VII.B.1.b(2), why *Browning et al.* does not teach this feature. The Office Action acknowledges (pp. 3-4) that *Armilli* does not teach this feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

c. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 1. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be overturned.

2. Independent Claim 8

a. The proposed combination does not teach "register by providing an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations. for the process

(1) RPN list processing in *Browning et al.* does not correspond to "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

The Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches…the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., ***scan all registered RPN lists and invalidates all entries, including virtual***

> ***address space, corresponding to real pages that within the range of memory to be removed***), and (col. 8 lines 53-58, i.e., ***receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed***).
> (Office Action, pp. 6-7, emphasis added.)

Appellant disagrees with this allegation.

The Office Action appears to assert that invalidating the RPN corresponds to an "indication…that the virtual address space is no longer available to the process", perhaps based on the presence of "an RPN data structure" in the relied-upon portion of FIG. 8 (reproduced above in section VII.B.1.a(1)) and the presence of a "virtual memory data structure" in claim 8. *Browning et al.* does disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN. (See FIG. 3A.) Appellant therefore assumes (for the sake of argument) that the RPN is a "virtual memory data structure". Even so, *Browning et al.* does not teach or suggest that invalidating an RPN entry provides an "indication…that the virtual address space is no longer available to the process" as recited in claim 8. Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page to get current, valid physical addresses.)

In addition, *Browning et al.* does not teach or suggest that the RPN is a data structure "for the process" as recited in claim 8. In fact, *Browning et al.* appears to suggest that the RPN is not associated with a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

(2) In-progress flag in *Browning et al.* does not correspond to "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

As discussed above, the Office Action specifically asserts that invalidating RPN in *Browning et al.* corresponds to the claimed indication. However, in the Response to Arguments section the Office Action changes position and appears to rely on clearing the memory-move-in-progress flag in *Browning et al.* for teaching the "for the process" portion of the indication:

> In respond to Appellant's argument that the proposed combination does not teach "provide an indication in a virtual memory data structure associated with the process", Browning clearly teaches providing a flag to indicate whether virtual memory is removed in progress (step 834, figure 8 and col. 8 lines 32-44) and reinitializing RPN entry and mark entry as valid when memory remove in progress flag is not set ***such that the memory remove in progress flag provides an indication in a virtual memory data structure associated with the process.***
> (Office Action, p. 12, emphasis added.)

Appellant disagrees with this allegation. First, *Browning et al.* does not teach or suggest that the memory-remove-in-progress flag is part of a virtual memory data structure. Second, *Browning et al.* does not teach or suggest that the flag has anything to do with a particular process as required by claim 8.

(3) *Browning et al.* as a whole does not teach "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

As noted above, the Office Action appears to take the position that updating one data structure (the memory-remove-in-progress flag) teaches one isolated portion ("indication in a virtual memory data structure that the virtual address space is no longer available to the process") of the claimed indication, and that updating a different data structure (the RPN entry) teaches the remaining part ("indication…that the virtual address space is no longer available to the process"). Yet the Examiner has provided no rationale for why these two separate data structures would be combined in the manner described in claim 8. The Examiner has failed to consider the claims as a whole. Appellant submits that it is not enough that all the individual elements of a claim are disclosed; those elements must also be arranged as in the claim.

(4) The combination as a whole does not teach "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

Appellant has explained, in sections VII.B.2.a(1)-VII.B.2.a(3), why *Browning et al.* does not teach this feature. The Office Action acknowledges (p. 6) that *Armilli* does not teach this feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

b. The proposed combination does not teach "to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations.

(1) Checking the memory-move-in-progress flag in *Browning et al.* does not correspond to "to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released"

The Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), ***when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released,*** and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed), and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs

and then triggering to register by detecting that real pages that are within range of memory to be removed).
(Office Action, p. 6, emphasis added.)

Appellant respectfully disagrees with this allegation.

The Office Action specifically refers to steps 824 and 834 of FIG. 8 for this teaching. With regard to step 824, Appellant respectfully submits that checking the validity of an RPN entry is not the same as detecting the release of physical addresses. Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid. (See FIG. 8, blocks 824 and 832.) Even assuming that checking validity of an RPN is the same as detecting release of physical addressees, the relied-upon portion of *Browning et al.* teaches that the action taken on detection is to update the virtual-to-physical mapping, rather than the registration recited in claim 8.

With regard to step 834, Appellant respectfully submits that checking the memory-remove-in-progress flag set flag in step 834 is not the same as detecting that the physical address space has been released. Instead, this flag merely indicates that process 900 (FIG. 9) is executing. Furthermore, claim 8 requires more than such a detection, it requires a specific action. The action that occurs in *Browning et al.* if the memory-remove-in-progress condition is detected is initialization of DMA mapping, not registration of any kind, much less the specific registration recited in claim 8, namely "register by providing indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process".

(2) In-progress flag in *Browning et al.* does not correspond to "to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released"

As discussed, above the Office Action specifically asserts that this feature corresponds to checking a particular condition (validity of RPN or the value of memory-move-in-progress flag) in *Browning et al.* However, in the Response to Arguments section the Office Action changes position and appears to rely instead on clearing the memory-move-in-progress flag:

> In respond to Appellant's argument that the proposed combination does not teach "wherein registering the indication is triggered by detection the physical address space…has been released", ***Browning teaches to clear memory remove in progress flag when it is acknowledge that real pages that are within range of memory has been removed or release*** (figure 9 and col. 8 line 65 through col. 9 line 10). Thus, Browning does teach the claimed limitations as recited in claim 1.
> (Office Action, p. 13, emphasis added.)

Appellant disagrees with this allegation.

First, Appellant submits that the "memory remove operation" referred to in FIG. 9 is not the same as the release of physical address space as described in claim 8. Instead, the "remove" is actually a relocate from one physical address to another. This is made clear in another portion of *Browning et al.*:

> In this manner, many different copies of a translation of a virtual address to physical address may exist. Some of these pretranslations may be invalidated, such as when the ***physical address changes because a real page is migrated to a new real page.*** When this occurs, the pretranslations to the original real page are invalid. The present invention provides a method, system, and product for locating particular pretranslations, invalidating them, synchronizing the invalidation process with the memory remove process, and then repopulating these lists with the current, valid pretranslation.
> (Col. 3, lines 17-25.)
>
> In order to synchronize invalidation of pretranslations stored in these lists with a memory remove operation, a user of a pretranslation list first disables the user's processor's ability to respond to interprocessor interrupts. The user then accesses the list. Once the user has finished accessing the list, the user then re-enables the ability of its processor to respond to interprocessor interrupts. Thus, while the user is accessing a pretranslation list, the user's processor will not respond to interprocessor interrupts.
> (Col. 3, lines 35-45.)

The synchronization using interprocessor interrupts described in this passage is the same synchronization described in connection with FIG. 9, leading to the conclusion that the migrate-real-page operation in the above passage is the same as the memory-remove operation of FIG. 9. Thus, the relied-upon passage (FIG. 9 and corresponding text) does not teach "physical address space…has been released", but rather that it has been moved.

Second, even assuming that the relied-upon passage teaches removing physical address space, it does not teach that this physical address space "was being used by processes

associated with the device" as required by claim 8. FIG. 9 of *Browning et al.* does not describe processes or a particular device in the context of moving physical pages.

Third, even assuming that detecting the kernel has completed migration of real pages teaches detecting that physical address space has been released, it does not teach that such detection triggers registration of any kind of any kind. According to FIG. 8, the action that occurs when the memory-remove-in-progress flag is clear (step 834) is marking the RPN entry as valid and reinitializing the entry (step 836). Appellant submits that changing values in an RPN entry is not the same as triggering a registration of kind, much less the specific registration recited in claim 8, namely "register by providing an…indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process".

(3) The combination as a whole does not teach "an indication in a virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for use by the process"

Appellant has explained, in sections VII.B.1.b(1) and VII.B.1.b(2), why *Browning et al.* does not teach this feature. The Office Action acknowledges (p. 6) that *Armilli* does not teach this feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

c. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 8. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be overturned.

3. Independent Claim 13

a. The proposed combination does not teach "means for unmapping a virtual address space for the process…in a manner which does not violate semantics for an operating system of the computing device"

The Office Action alleges that *Armilli* teaches this feature as follows:

> means for un-mapping a virtual address space, i.e., processor's move engine (28, figure 3), for a process in a manner which does not violate semantics for an operating system of the computing device when a removable memory mappable device associated with the process is logically disconnected (abstract and col. 7 line 32 through col. 9 line 10, i.e., the processor's move engine works in conjunction with the associated mapping engine to take the associated memory module offline, read as un-mapping a virtual address space, prior to its physically removal, read as when the removable memory mappable device associated with the process is logically disconnected, ***such that the memory module can be removed in physical memory without the operating system having to direct and control the reconfiguration of physical memory to accomplish the physical memory chance, read as for a process in a manner which does not violate semantics for an operating system of the computing device***). (Office Action, pp. 8-9, emphasis added.)

Thus, the Office Action interprets the feature "in a manner which does not violate semantics for an operating system of the computing device" as corresponding to "without the operating system having to direct and recontrol the reconfiguration of physical memory". Appellant submits this interpretation is incorrect because unmapping without a need for operating system supervision is not what is claimed. Instead, claim 13 refers to operating system semantics, and the Office Action has not explained why a person of ordinary skill in the art would understand supervision and semantics to be the same.

More information about techniques for "unmapping…in a manner which does not violate semantics for an operating system of the computing device" can be found in the following passages in the instant specification:

> To achieve this, the program instructions execute to ***unmap the virtual address space in a manner which does not violate semantics for an operating system of the computing device***, e.g., a computing device having a Unix operating system. For example, referring to Figure 3, in various embodiments the program instructions execute to maintain a representation of an object associated with the process in the virtual memory data structure of the process, e.g., in the pregion data structure shown in Figure 3. Meanwhile, the program instructions can execute to remove a mapping of the object to physical memory. Additionally, the program instructions execute to register in the virtual

memory data structure of the process, e.g., shown in Figure 3, that the virtual address space associated with the process is not available for use. By way of example and not by way of limitation, the program instructions execute to set a bit in a region, e.g., 308 in Figure 3, of the virtual memory data structure to indicate that the virtual address space in not available for use. In this manner, the program instructions execute to indicate an operation as failed if the process attempts to perform the operation subsequent to registering that the virtual address space is no longer valid for process use. And, the program instructions can execute to allow the process to unmap the virtual address space subsequent to the release of the physical address space.
(p. 17, lines 1-19.)

The program instructions execute to mark a location within a virtual memory data structure associated with a given process that the virtual address space allocated to the process is no longer available for use when a memory mappable device associated with that process is logically removed, e.g., powered off, physically removed, or otherwise. And, the program instructions execute to maintain a representation of an object, e.g., block of data, text or graphics, that was created by that process and had a virtual address space allocated to it in the virtual memory data structure as well. By providing such an indication, the program instructions described herein can execute to indicate an operation as failed if the process attempts to perform the operation subsequent to the memory mappable device being logically disconnected from the computing device. And, the program instructions associated with the process can in their regular manner execute to release the particular allocated virtual address space at the process's request subsequent to the memory mappable device being logically disconnected from the computing device. Thus, according to the method embodiments described herein program instructions execute to effectively ***unmap a virtual address space associated with a process without violating the semantics of an operating system of a computing device.***
(p. 19, lines 6-23.)

Ordinarily, this would potentially permit multiple processes to begin to conflict and foul one another up by reading and writing data into a virtual memory address space which is not intended to be shared. As shown in block 530, however, program embodiments of the present invention operate to prevent this from occurring by executing to ***register that the virtual address space is not available to the process, or processes (e.g., associated with the particular removable device) in a manner which does not violate semantics of a given operating system***. The program embodiments can execute to register that the virtual address space is not available to the process, or processes, according to any of the methods discussed and described above in connection with Figure 4. As described in connection with Figure 4, the program instructions which execute to register the virtual address space is not available additionally execute to maintain a representation of an object that was created by that process and had a virtual address space allocated to it in the virtual memory data structure. As one of ordinary skill in the art will appreciate upon reading this disclosure the above program instructions can be triggered to execute upon detection that the operating system has released a physical address space which was being used

> by processes associated with a particular removable device that had been mapped to memory using a virtual address scheme.
> (p. 20 line 17 to p. 21 line 2, emphasis added.)

Appellant therefore submit that *Armilli* does not disclose unmapping a virtual address space in a manner which does not violate operating system semantics, and *Browning et al.* does not cure this deficiency. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

4. Independent Claim 19

a. The proposed combination does not teach "providing an indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations.

(1) RPN list processing in *Browning et al.* does not correspond to "indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process"

The Office Action indicates (p. 11) that claim 19 is rejected for the same reasons as claim 8. In rejecting claim 8, the Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches…the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., ***scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed***), and (col. 8 lines 53-58, i.e., ***receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed***).
> (Office Action, p. 9, emphasis added.)

Appellant disagrees with this allegation.

Appellant notes that the relied-upon portion of FIG. 8 (reproduced above in section VII.B.1.a(1)) uses the term "valid" (steps 824, 836, and 914) when referring to an RPN data structure. Appellant also notes that the claim refers to a "virtual memory data structure" and includes the functional language "that a virtual address space is no longer available for use by the process". Thus, the Office Action appears to rely on the RPN for teaching the claimed "virtual memory data structure".

Browning et al. does disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN. (See FIG. 3A.) Appellant therefore assumes (for the sake of argument) that the RPN is a "virtual memory data structure". Even so, *Browning et al.* does not teach or suggest that invalidating an RPN entry provides an "indication...that a virtual address space is no longer available for use by the process" as recited in claim 19. Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page to get current, valid physical addresses.)

In addition, *Browning et al.* does not teach or suggest that the RPN is a data structure "for the process" as recited in claim 19. In fact, *Browning et al.* appears to suggest that the RPN is not associated with a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

(2) In-progress flag in *Browning et al.* does not correspond to "indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process"

As discussed above, the Office Action specifically asserts that invalidating the RPN in *Browning et al.* corresponds to the claimed indication. However, in the Response to Arguments

section the Office Action changes position and appears to rely on clearing the memory-move-in-progress flag in *Browning et al.* for teaching the "for the process" portion of the indication:

> In respond to Appellant's argument that the proposed combination does not teach "provide an indication in a virtual memory data structure associated with the process", Browning clearly teaches providing a flag to indicate whether virtual memory is removed in progress (step 834, figure 8 and col. 8 lines 32-44) and reinitializing RPN entry and mark entry as valid when memory remove in progress flag is not set ***such that the memory remove in progress flag provides an indication in a virtual memory data structure associated with the process.***
> (Office Action, p. 12, emphasis added.)

Appellant disagrees with this allegation. First, *Browning et al.* does not teach or suggest that the memory-remove-in-progress flag is part of a virtual memory data structure. Second, *Browning et al.* does not teach or suggest that the flag has anything to do with a particular process as required by claim 19.

(3) *Browning et al.* as a whole does not teach "indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process"

As noted above, the Office Action appears to take the position that updating one data structure (the memory-remove-in-progress flag) teaches one isolated portion ("indication in a virtual memory data structure for the process") of the claimed indication, and that updating a different data structure (the RPN entry) teaches the remaining part ("indication…that a virtual address space is no longer available for use by the process"). Yet the Examiner has provided no rationale for why these two separate data structures would be combined in the manner described in claim 19. The Examiner has failed to consider the claims as a whole. Appellant submits that it is not enough that all the individual elements of a claim are disclosed; those elements must also be arranged as in the claim.

(4) The combination as a whole does not teach "indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process"

Appellant has explained, in sections VII.B.4.a(1)-VII.B.4.a(3), why *Browning et al.* does not teach this feature. The Office Action acknowledges (pp. 3-4) that *Armilli* does not teach this

feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

b. The proposed combination does not teach "as triggered by detection of a physical address space used by the process being released and when the object is removed from physical memory"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations.

(1) Checking the memory-move-in-progress flag in *Browning et al.* does not correspond to "as triggered by detection of a physical address space used by the process being released and when the object is removed from physical memory"

The Office Action indicates (p. 11) that claim 19 is rejected for the same reasons as claim 8. In rejecting claim 8, the Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), ***when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released,*** and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed), and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Appellant respectfully disagrees with this allegation.

The Office Action specifically refers to steps 824 and 834 of FIG. 8 for this teaching. With regard to step 824, Appellant respectfully submits that checking the validity of an RPN entry is not the same as detecting the release of physical addresses. Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid. (See FIG. 8, blocks 824 and 832.) Even assuming that checking validity of an RPN is the same as detecting release of physical addressees, the relied-upon portion of *Browning et al.* teaches that the action taken on detection is to update the virtual-to-physical mapping, rather than to trigger an indication as recited in claim 19.

With regard to step 834, Appellant respectfully submits that checking the memory-remove-in-progress flag set flag in step 834 is not the same as detecting that the physical address space has been released. Instead, this flag merely indicates that process 900 (FIG. 9) is executing. Furthermore, claim 19 requires more than such a detection, it requires a specific action. The action that occurs in *Browning et al.* if the memory-remove-in-progress condition is detected is initialization of DMA mapping, not triggering an indication of any kind, much less the specific indication recited in claim 19, namely an "indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process".

(2) In-progress flag in *Browning et al.* does not correspond to "as triggered by detection of a physical address space used by the process being released and when the object is removed from physical memory"

As discussed above, the Office Action specifically asserts that this feature corresponds to checking a particular condition (validity of RPN or the value of memory-move-in-progress flag) in *Browning et al.* However, in the Response to Arguments section the Office Action changes position and appears to rely instead on clearing the memory-move-in-progress flag:

> In respond to Appellant's argument that the proposed combination does not teach "wherein registering the indication is triggered by detection the physical address space…has been released", ***Browning teaches to clear memory remove in progress flag when it is acknowledge that real pages that are within range of memory has been removed or release*** (figure 9 and col. 8 line

> 65 through col. 9 line 10). Thus, Browning does teach the claimed limitations as recited in claim 1.
> (Office Action, p. 13, emphasis added.)

Appellant disagrees with this allegation.

First, Appellant submits that the "memory remove operation" referred to in FIG. 9 is not the same as the release of physical address space as described in claim 19. Instead, the "remove" is actually a migration from one physical address to another. This is made clear in another portion of *Browning et al.*:

> In this manner, many different copies of a translation of a virtual address to physical address may exist. Some of these pretranslations may be invalidated, such as when the ***physical address changes because a real page is migrated to a new real page.*** When this occurs, the pretranslations to the original real page are invalid. The present invention provides a method, system, and product for locating particular pretranslations, invalidating them, synchronizing the invalidation process with the memory remove process, and then repopulating these lists with the current, valid pretranslation.
> (Col. 3, lines 17-25.)
>
> In order to synchronize invalidation of pretranslations stored in these lists with a memory remove operation, a user of a pretranslation list first disables the user's processor's ability to respond to interprocessor interrupts. The user then accesses the list. Once the user has finished accessing the list, the user then re-enables the ability of its processor to respond to interprocessor interrupts. Thus, while the user is accessing a pretranslation list, the user's processor will not respond to interprocessor interrupts.
> (Col. 3, lines 35-45.)

The synchronization using interprocessor interrupts described in this passage is the same synchronization described in connection with FIG. 9, leading to the conclusion that the migrate-real-page operation in the above passage is the same as the memory-remove operation of FIG. 9. Thus, the relied-upon passage (FIG. 9 and corresponding text) does not teach "physical address space…being released", but rather being moved.

Second, even assuming that detecting the kernel has completed migration of real pages teaches detecting that physical address space has been released, it does not teach that such detection triggers an indication of any kind. According to FIG. 8, the action that occurs when the memory-remove-in-progress flag is clear (step 834) is marking the RPN entry as valid and reinitializing the entry (step 836). Appellant submits that changing values in an RPN entry is not

the same as triggering an indication of kind, much less the specific indication recited in claim 19, namely an "indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process".

Third, even assuming that the relied-upon passage teaches removing physical address space, it does not teach that this physical address space "used by the process" as required by claim 19. FIG. 9 of *Browning et al.* does not describe processes in the context of moving physical pages.

(3) The combination as a whole does not teach "an indication in a virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for use by the process"

Appellant has explained, in sections VII.B.4.b(1) and VII.B.4.b(2), why *Browning et al.* does not teach this feature. The Office Action acknowledges (pp. 3-4) that *Armilli* does not teach this feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

c. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 19. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be overturned.

5. Independent Claim 22

a. The proposed combination does not teach "registering an indication in a virtual memory data structure for the process that the virtual address space is not available to the process"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations.

(1) RPN list processing in *Browning et al.* does not correspond to "indication in a virtual memory data structure for the process that the virtual address space is not available to the process"

The Office Action indicates (p. 11) that claim 22 is rejected for the same reasons as claim 1. In rejecting claim 1, the Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches…the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., ***scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed***), and (col. 8 lines 53-58, i.e., ***receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed***).
> (Office Action, pp. 3-4, emphasis added.)

Appellant disagrees with this allegation.

The Office Action appears to assert that invalidating the RPN corresponds to an "indication…that the virtual address space is no longer available to the process", perhaps based on the presence of "an RPN data structure" in the relied-upon portion of FIG. 8 (reproduced in section VII.B.1.a(1)) and the presence of a "virtual memory data structure" in claim 22. *Browning et al.* does disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN. (See FIG. 3A.) Appellant therefore assumes (for the sake of argument) that the RPN is a "virtual memory data structure". Even so, *Browning et al.* does not teach or suggest that invalidating an RPN entry provides an "indication…that the virtual address space is not available to the process" as recited in claim 22. Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page to get current, valid physical addresses.)

In addition, *Browning et al.* does not teach or suggest that the RPN is a data structure "for the process" as recited in claim 22. In fact, *Browning et al.* appears to suggest that the RPN is not associated with a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

(2) In-progress flag in *Browning et al.* does not correspond to "indication in a virtual memory data structure for the process that the virtual address space is not available to the process"

As discussed above, the Office Action specifically asserts that invalidating RPN in *Browning et al.* corresponds to the claimed indication. However, in the Response to Arguments section the Office Action changes position and appears to rely on clearing the memory-move-in-progress flag in *Browning et al.* for teaching the "for the process" portion of the indication:

> In respond to Appellant's argument that the proposed combination does not teach "provide an indication in a virtual memory data structure associated with the process", Browning clearly teaches providing a flag to indicate whether virtual memory is removed in progress (step 834, figure 8 and col. 8 lines 32-44) and reinitializing RPN entry and mark entry as valid when memory remove in progress flag is not set ***such that the memory remove in progress flag provides an indication in a virtual memory data structure associated with the process.***
> (Office Action, p. 12, emphasis added.)

Appellant disagrees with this allegation. First, *Browning et al.* does not teach or suggest that the memory-remove-in-progress flag is part of a virtual memory data structure. Second, *Browning et al.* does not teach or suggest that the flag has anything to do with a particular process as required by claim 22.

(3) *Browning et al.* as a whole does not teach "indication in a virtual memory data structure for the process that the virtual address space is not available to the process"

As noted above, the Office Action appears to take the position that updating one data structure (the memory-remove-in-progress flag) teaches one isolated portion ("indication in a virtual memory data structure for the process") of the claimed indication, and that updating a different data structure (the RPN entry) teaches the remaining part ("indication…that the virtual address space is not available to the process"). Yet the Examiner has provided no rationale for why these two separate data structures would be combined in the manner described in claim 22. The Examiner has failed to consider the claims as a whole. Appellant submits that it is not enough that all the individual elements of a claim are disclosed; those elements must also be arranged as in the claim.

(4) The combination as a whole does not teach "indication in a virtual memory data structure for the process that the virtual address space is not available to the process"

Appellant has explained, in sections VII.B.5.a(1)-VII.B.5.a(3), why *Browning et al.* does not teach this feature. The Office Action acknowledges (pp. 3-4) that *Armilli* does not teach this feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

b. The proposed combination does not teach "[registering an indication] at the release of the **physical address space used by the process and before the process has released the virtual address space"**

In the Response to Arguments section, Office Action alleges that *Browning et al.* teaches this feature as follows:

> Examiner took a broadly interpretation to make a rejection based upon the other similar limitation in claims 13, 19,22, and 23, which registering is triggered to detect the physical address space is released, when the device is being released. According, Browning teaches to determining memory removal in process flag set, i.e., memory is being released or not, after confirmed the RPN entry for the virtual memory page valid by the DMA mapping engine, and if the flag is set, it further perform DMA mapping to remapping corresponding to the

> virtual memory page, i.e., old physical address space is released before release of the virtual address space, as defined in figure 8, and col. 7 line 21 through col. 8 line 64),.
> (Office Action, p. 12.)

As Appellant can best decipher, this remark in the Office Action amounts to an allegation that:

> (1) the memory removal in progress flag being set (Yes path at block 834 in FIG. 8) corresponds to "old physical address space has been released"
> (2) initialization of the DMA mapping to point to a physical address (block 828 in FIG. 8) corresponds to "release of the virtual address space".

The Office Action appears to further allege that (1) happens before (2), so that the two teachings together (allegedly) amount to "old physical address space has been released occurs before release of the virtual address space".

Appellant first disagrees with point 1, as to the meaning of the memory removal in progress flag being set. The clear teaching of FIG. 9 of *Browning et al.* is that a memory migration of real pages is in process when the flag is set. However, Appellant also submits that a clear memory removal in progress flag does not mean that the pages have been removed, since the initial state of the flag (before the process of FIG. 9 has ever run once) is clear. Thus, Appellant submits that the state of the memory remove in progress flag cannot serve as a trigger for the release of physical address space. Instead, the flag is simply a mechanism for the process of FIG. 8 to avoid touching the RPN entries in block 835 while FIG. 9 is executing (and also touching the RPN entries).

Appellant also disagrees with point 2, as to the meaning of initializing a DMA mapping in step 828 of FIG. 8. Step 828 clearly states that pages of a particular virtual memory buffer, requested in step 802, are mapped to physical pages. Appellant fails to see how this mapping can correspond to the release of a virtual address space. To the contrary, Appellant submits a virtual page mapped to a physical page is in-use – the opposite of released.

Appellant submits that *Browning et al.* does contain a teaching that is relevant to the timing of invalidation of RPNs and memory migration. Specifically, the description of FIG. 9 at Col. 8, line 45 to Col. 9, line 10 appears to teach a kernel scanning all registered RPN lists and

invalidating all entries that correspond to real pages that are within the range of memory to be moved, and finally performs memory migration and removal of real pages of memory. Thus, the kernel invalidates the relevant pretranslation lists before memory migration of physical pages of memory. Therefore, to the extent that invalidation of RPN entries corresponds to marking a virtual address as not in use and migration of physical pages corresponds to release of physical address space, *Browning et al.* teaches that this migration occurs after the virtual address is marked as not in use – in direct contradiction to the characterization of *Browning et al.* found at p. 12 of the Office Action.

Finally, Appellant submits that even if the characterization of *Browning et al.* expressed in points 1 and 2 was accurate, it is not enough that *Browning et al.* teaches "old physical address space has been released occurs before release of the virtual address space" since claim 23 does not recite this feature in isolation, Instead, this "at the release…and before…" feature describes the timing of a particular action, namely "registering an indication". As argued extensively above (in sections VII.B.5.a), *Browning et al.* simply does not disclose this action.

c. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 22. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be overturned.

6. Independent Claim 23

a. The proposed combination does not teach "registering an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

The Office Action alleges that *Browning et al.* teaches this feature. Specifically, the Office Action appears to make two different allegations about how *Browning et al.* teaches this feature. Appellant now addresses each of those allegations.

(1) RPN list processing in *Browning et al.* does not correspond to "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

The Office Action indicates (p. 11) that claim 23 is rejected for the same reasons as claim 19, and also indicates that claim 19 was rejected for the same reasons as claim 8. In rejecting claim 8, the Office Action first alleges in the main body of the rejection that *Browning et al.* teaches this feature as follows:

> However, Browning teaches…the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., ***scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed***), and (col. 8 lines 53-58, i.e., ***receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed***).
> (Office Action, pp. 6-7, emphasis added.)

Appellant disagrees with this allegation.

The Office Action appears to assert that invalidating the RPN corresponds to an "indication…that the virtual address space is no longer available to the process", perhaps based on the presence of "an RPN data structure" in the relied-upon portion of FIG. 8 (reproduced in section VII.B.1.a) and the presence of a "virtual memory data structure" in claim 23. *Browning et al.* does disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN. (See FIG. 3A.) Appellant therefore assumes (for the sake of argument) that the RPN is a "virtual memory data structure". Even so, *Browning et al.* does not teach or suggest that invalidating an RPN entry provides an "indication…that the virtual address space is no longer available to the process" as recited in claim 23. Instead, an invalid RPN entry merely signifies that the ***virtual-to-physical mapping*** contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page to get current, valid physical addresses.)

In addition, *Browning et al.* does not teach or suggest that the RPN is a data structure "for the process" as recited in claim 23. In fact, *Browning et al.* appears to suggest that the RPN is not associated with a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

(2) In-progress flag in *Browning et al.* does not correspond to "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

As discussed above, the Office Action specifically asserts that invalidating RPN in *Browning et al.* corresponds to the claimed indication. However, in the Response to Arguments section the Office Action changes position and appears to rely on clearing the memory-move-in-progress flag in *Browning et al.* for teaching the "for the process" portion of the indication:

> In respond to Appellant's argument that the proposed combination does not teach "provide an indication in a virtual memory data structure associated with the process", Browning clearly teaches providing a flag to indicate whether virtual memory is removed in progress (step 834, figure 8 and col. 8 lines 32-44) and reinitializing RPN entry and mark entry as valid when memory remove in progress flag is not set ***such that the memory remove in progress flag provides an indication in a virtual memory data structure associated with the process.***
> (Office Action, p. 12, emphasis added.)

Appellant disagrees with this allegation. First, *Browning et al.* does not teach or suggest that the memory-remove-in-progress flag is part of a virtual memory data structure. Second, *Browning et al.* does not teach or suggest that the flag has anything to do with a particular process as required by claim 23.

(3) *Browning et al.* as a whole does not teach "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

As noted above, the Office Action appears to take the position that updating one data structure (the memory-remove-in-progress flag) teaches one isolated portion ("indication in a virtual memory data structure for the process") of the claimed indication, and that updating a different data structure (the RPN entry) teaches the remaining part ("indication…that the virtual address space is no longer available to the process"). Yet the Examiner has provided no rationale for why these two separate data structures would be combined in the manner described in claim 23. The Examiner has failed to consider the claims as a whole. Appellant submits that it is not enough that all the individual elements of a claim are disclosed; those elements must also be arranged as in the claim.

(4) The combination as a whole does not teach "indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process"

Appellant has explained, in sections VII.B.1.a(1)-VII.B.1.a(3), why *Browning et al.* does not teach this feature. The Office Action acknowledges (p. 5) that *Armilli* does not teach this feature. Since the references do not (individually or in combination) disclose, teach, or suggest this feature, a *prima facie* case of obviousness has not been made, and the rejection should be overturned.

b. The proposed combination does not teach "[registering an indication] at the release of the **physical address space used by the process and before the process has released the virtual address space"**

In the Response to Arguments section, Office Action alleges that *Browning et al.* teaches this feature as follows:

> Examiner took a broadly interpretation to make a rejection based upon the other similar limitation in claims 13, 19,22, and 23, which registering is triggered to detect the physical address space is released, when the device is being released. According, Browning teaches to determining memory removal in process flag set, i.e., memory is being released or not, after confirmed the RPN entry for the virtual memory page valid by the DMA mapping engine, and if the flag is set, it further perform DMA mapping to remapping corresponding to the

> virtual memory page, i.e., old physical address space is released before release of the virtual address space, as defined in figure 8, and col. 7 line 21 through col. 8 line 64),.
> (Office Action, p. 12.)

As Appellant can best decipher, this remark in the Office Action amounts to an allegation that:

> (1) the memory removal in progress flag being set (Yes path at block 834 in FIG. 8) corresponds to "old physical address space has been released"
> (2) initialization of the DMA mapping to point to a physical address (block 828 in FIG. *) corresponds to "release of the virtual address space".

The Office Action appears to further allege that (1) happens before (2), so that the two teachings together (allegedly) amount to "old physical address space has been released occurs before release of the virtual address space".

Appellant first disagrees with point 1, as to the meaning of the memory removal in progress flag being set. The clear teaching of FIG. 9 of *Browning et al.* is that a memory migration of real pages is in process when the flag is set. However, Appellant also submits that a clear memory removal in progress flag does not mean that the pages have been removed, since the initial state of the flag (before the process of FIG. 9 has ever run once) is clear. Thus, Appellant submits that the state of the memory remove in progress flag cannot serve as a trigger for the release of physical address space. Instead, the flag is simply a mechanism for the process of FIG. 8 to avoid touching the RPN entries in block 835 while FIG. 9 is executing (and also touching the RPN entries).

Appellant also disagrees with point 2, as to the meaning of initializing a DMA mapping in step 828 of FIG. 8. Step 828 clearly states that pages of a particular virtual memory buffer, requested in step 802, are mapped to physical pages. Appellant fails to see how this mapping can correspond to the release of a virtual address space. To the contrary, Appellant submits a virtual page mapped to a physical page is in-use – the opposite of released.

Appellant submits that *Browning et al.* does contain a teaching that is relevant to the timing of invalidation of RPNs and memory migration. Specifically, the description of FIG. 9 at Col. 8, line 45 to Col. 9, line 10 appears to teach a kernel scanning all registered RPN lists and

invalidating all entries that correspond to real pages that are within the range of memory to be moved, and finally performs memory migration and removal of real pages of memory. Thus, the kernel invalidates the relevant pretranslation lists before memory migration of physical pages of memory. Therefore, to the extent that invalidation of RPN entries corresponds to marking a virtual address as not in use and migration of physical pages corresponds to release of physical address space, *Browning et al.* teaches that this migration occurs after the virtual address is marked as not in use – in direct contradiction to the characterization of *Browning et al.* found at p. 12 of the Office Action.

Finally, Appellant submits that even if the characterization of *Browning et al.* expressed in points 1 and 2 were accurate, it is not enough that *Browning et al.* teaches "old physical address space has been released occurs before release of the virtual address space" since claim 23 does not recite this feature in isolation, Instead, this "at the release…and before…" feature describes the timing of a particular action, namely "registering an indication". As argued extensively above (in sections VII.B.5.a), *Browning et al.* simply does not disclose this action.

c. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 23. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be overturned.

7. Dependent Claims 2-7, 9-12, 14-18, and 20-21

Since independent claims 1, 8, 13, 19, 22, and 23 are allowable, Appellant respectfully submits that claims 2-7, 9-12, 14-18, and 20-21 are allowable for at least the reason that each depends from an allowable claim. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596, 1598 (Fed. Cir. 1988). Therefore, Appellant respectfully requests that the rejection of claims 2-7, 9-12, 14-18, and 20-21 be overturned.

C. Conclusion

For at least the reasons discussed above, Appellant respectfully requests that the Examiner's rejection of claims 1-23 be overturned by the Board. In addition to the claims listed in Section VIII (CLAIMS – APPENDIX), Section IX (EVIDENCE – APPENDIX) included herein indicates that there is no additional evidence relied upon by this brief. Section X (RELATED PROCEEDINGS – APPENDIX) included herein indicates that there are no related proceedings.

Respectfully submitted,

By: /Karen G. Hazzah/

Karen G. Hazzah,
Reg. No. 48,472

THOMAS, KAYDEN, HORSTEMEYER & RISLEY, L.L.P.
600 Galleria Parkway, NW
Suite 1500
Atlanta, Georgia 30339-5948
Tel: (770) 933-9500
Fax: (770) 951-0933

VIII. CLAIMS – APPENDIX

1. A computing device, comprising:

a processor;

a memory coupled to the processor; and

program instructions provided to the memory and executable by the processor to:

track a virtual address space for a process associated with a device connected to the computing device;

release a physical address space associated with the virtual address space when the device has a connection removed from the computing device;

provide an indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid for use by the process;

wherein the indication is triggered by detection that the physical address space that was being used by processes associated with the device has been released; and

wherein the indication occurs responsive to the physical address space being released and before release of the virtual address space by the process.

2. The computing device of claim 1, wherein the device includes a device which can be mapped to memory.

3. The computing device of claim 1, wherein the virtual address space includes an input/output space.

4. The computing device of claim 1, wherein the program instructions are part of a memory management system which includes a virtual memory data structure associated with the process.

5. The computing device of claim 4, wherein the program instructions execute to register the virtual address space is no longer valid for process use in the virtual memory data structure associated with the process.

6. The computing device of claim 1, wherein the program instructions execute to allocate the virtual address space when the process requests physical memory.

7. The computing device of claim 1, wherein the program instructions execute to register that the virtual address space is available for use when the process releases the virtual address space.

8. A computing device, comprising:

a processor;

a random access memory coupled to the processor; and

program instructions provided to the memory and executable by the processor, the program instructions are part of a memory management system to:

dereference a virtual address space for a process associated with a removable memory mappable device connected to the computing device;

release a physical address space associated with the virtual address space when the device associated with the process is logically disconnected; and

register by providing an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process;

wherein to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released; and

wherein to register occurs as the physical address space is released and before release of the virtual address space by the process.

9. The computing device of claim 8, wherein the program instructions execute to unmap the virtual address space in a manner which do not violate semantics for an operating system of the computing device.

10. The computing device of claim 9, wherein the operating system is selected from the group of a Unix operating system and a Linux operating system.

11. The computing device of claim 8, wherein the program instructions execute to allow the process to unmap the virtual address space subsequent to the release of the physical address space.

12. The computing device of claim 8, wherein the program instructions execute to indicate an operation as failed if the process attempts to perform the operation subsequent to registering that the virtual address space is no longer valid for process use.

13. A computing device, comprising:

a processor;

a memory coupled to the processor, the memory including program instructions for maintaining a virtual memory data structure specific to a process as part of a memory management system; and

means for unmapping a virtual address space for the process that is triggered as a physical address space used by the process is being released, in a manner which does not violate semantics for an operating system of the computing device, when a removable memory mappable device associated with the process is logically disconnected.

14. The computing device of claim 13, wherein the program instructions execute to dereference the virtual address space for the process.

15. The computing device of claim 13, wherein the means for unmapping the virtual address space includes program instructions which execute to maintain a representation of an object associated with the process in the virtual memory data structure of the process.

16. The computing device of claim 15, wherein the means for unmapping the virtual address space includes program instructions which execute to remove a mapping of the object to physical memory.

17. The computing device of claim 16, wherein the means for unmapping the virtual address space includes program instructions which execute to register in the virtual memory data structure of the process that the virtual address space associated with the process is not available for use subsequent to when the mapping of the object to physical memory has been removed.

18. The computing device of claim 17, wherein the program instructions execute to set a bit in a pregion of the virtual memory data structure to indicate that the virtual address space is not available for use.

19. A method for memory management on a computing device, comprising:

dereferencing a memory address for a process associated with a removable memory mappable device;

mapping a representation of an object associated with the process in a virtual memory data structure associated with the process;

removing the object from physical memory when the device is logically disconnected from the computing device; and

providing an indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process as triggered by detection of a

physical address space used by the process being released and when the object is removed from physical memory, without removing the representation of the object from the virtual memory data structure for the process.

20. The method of claim 19, further including unmapping the virtual address space at the request of the process subsequent to the device being logically disconnected from the computing device.

21. The method of claim 19, further including indicating an operation as failed if the process attempts to perform the operation subsequent the device being logically disconnected from the computing device.

22. A method for memory management, comprising:

tracking a virtual address space for a process associated with a removable memory mappable device connected to a computing device;

releasing a physical address space when the device has a logical connection removed from the computing device; and

at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is not available to the process in a manner which does not violate semantics of an operating system.

23. A computer readable storage medium having computer readable instructions stored thereon for execution by a device to perform a method, comprising:

dereferencing a virtual address space for a process associated with a removable memory mappable device as part of a memory management system on a computing device;

releasing a physical address space when the device is logically disconnected from the computing device; and

at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process in a manner which does not violate semantics for an operating system the computing device.

IX. EVIDENCE – APPENDIX

None.

X. RELATED PROCEEDINGS – APPENDIX

None.